

Verwaltung großer Datenmengen auf einem Netzwerklaufwerk. Ein Vergleich zwischen MS-ACCESS und SQLite.

Was tun, wenn man als Entwickler eine Datenbank einsetzen will, aber keine SQL-Datenbank installieren darf. Eine Lösung sind dateibasierte Datenbanken. Der vorliegende Artikel vergleicht zwei dateibasierte Datenbanken (MS-Access und SQLite) hinsichtlich der Einsetzbarkeit auf einem Netzwerklaufwerk.

Einleitung

Eine dateibasierte Datenbank ist "nur" eine Datei, die auf einem (Netzwerk-) Laufwerk liegt. Auf diese Datei wird von mehreren Client-Programmen lesend und schreibend zugegriffen. Wenn ein Client in der Datenbank anfragt, werden alle für die Abfrage relevanten Tabellen an den Client übertragen. Die Datenverarbeitung erfolgt dann ausschließlich auf dem Client.

Im Unterschied dazu wird ein SQL-Datenbankmanagementsystem, welches SQL-Datenbank(en) beinhaltet, auf einem Server installiert. Das Datenbankmanagementsystem übernimmt dabei u.a. folgende Aufgaben: Es verarbeitet serverseitig die Anfragen der Clients und gibt nur die Daten zurück, die der Client anfragt. Zudem steuert es den Multi-User Zugriff und das Rechtemanagement. Die Datenverarbeitung erfolgt in der Regel auf dem Server.

Eine SQL-Datenbank ist deshalb in der Regel immer die optimale Lösung zur Verwaltung größerer Datenmengen. In bestimmten Szenarien kann es aber sein, dass eine SQL-Datenbank nicht installiert werden darf/kann. Dies kann z.B. der Fall sein, wenn ein externer Consultant/Dienstleister bei einem Kunden nur über eine Netzwerkfreigabe verfügt.

Ziel war es, den Einsatzbereich dateibasierter Datenbanken auf einem Netzwerklaufwerk zu untersuchen. Das Szenario gibt vor, dass keine Software auf dem Server installiert werden darf. Eine clientseitige Installation soll ebenfalls soweit wie möglich vermieden werden. Ein SQL-Datenbankmanagementsystem scheidet deshalb als Lösungsansatz aus. Als Lösungsstrategie kommen somit nur filebasierte Datenbanken in Betracht.

Die Datenbanksysteme

Als Datenbanksysteme, wurden in unserem Szenario MS-Access und SQLite ausgewählt. Es muss erwähnt werden, dass es sich um ein exemplarisches TestszENARIO handelt. In anderen Hardwareumgebungen können andere Ergebnisse erzielt werden. Die Grundaussagen sind unserer Ansicht nach aber allgemein gültig. Die folgenden Abschnitte gehen auf die Leistung und auf die Vor- und Nachteile der beiden Systeme ein.

Leistung

Als erstes wurde eine kleine Testdatenbank für beide Datenbanksysteme erstellt. Darin enthalten sind die Tabellen für Kunden, Artikel, Aufträge und Auftragspositionen. Jede dieser Tabellen beinhaltet fünf bis sechs Spalten. Insgesamt sind 59.769 Datensätze in jeder der beiden Datenbanken enthalten (verteilt über die Tabellen).

Auf diese beiden Datenbanken wurde folgende SQL Abfrage angewendet:

```

1 SELECT DISTINCT AuftragPositionNr, Auftrag.AuftragNr, Artikel.ArtikelNr,
2     Kunde.KundenNr, Artikel.Artikelname, Artikel.Artikelkurzname, Artikel.TOPCON,
3     Fertigstellungstermin, Liefertermin, Kunde.Kundenname1, Kunde.Kundensuchname
4 FROM Kunde, Artikel, Auftrag, AuftragPosition
5 WHERE AuftragPosition.AuftragNr = Auftrag.AuftragNr AND
6     AuftragPosition.ArtikelNr = Artikel.ArtikelNr AND
7     Auftrag.KundeNr = Kunde.KundenNr
8 ORDER BY Kunde.Kundenname1

```

Folgendes Ergebnis wurde dabei erzielt:

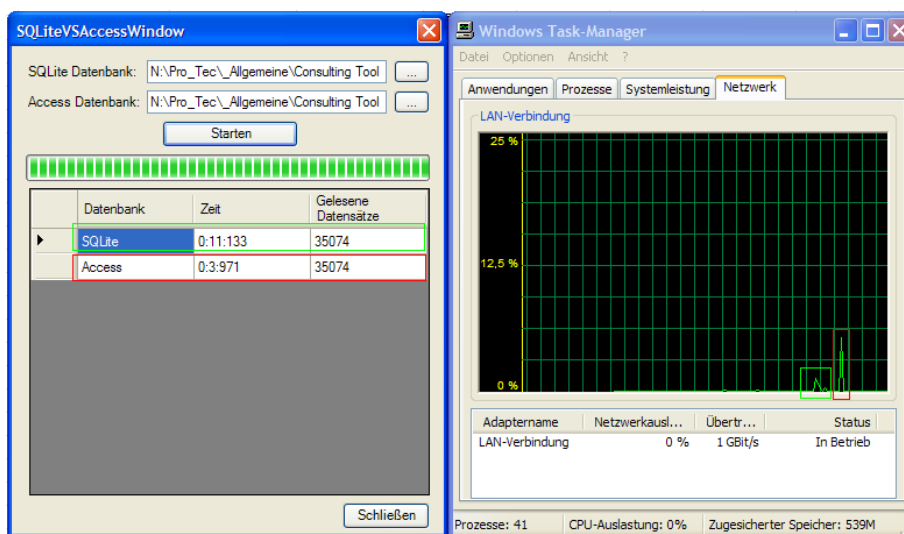


Abbildung 1: Leistung von SQLite und Access im Vergleich

In Abbildung 1 sieht man, dass Access (rot markiert), mit 3 Sekunden und 971 Millisekunden gegenüber SQLite (grün markiert), mit 11 Sekunden und 133 Millisekunden, eindeutig die Nase vorn hat. Wie man an der Netzwerkauslastung (siehe Abbildung 1) erkennen kann, hat SQLite länger benötigt um die Daten von dem Netzwerklaufwerk zu erhalten, hat das Netzwerk aber auch weniger belastet.

Zusätzlich wurde eine Testdatenbank generiert, welche 5 Tabellen besitzt. Jede Tabelle besitzt 51 Spalten (eine Spalte für den Primärschlüssel, eine Spalte für eine Zahl und der Rest der Spalten für Zeichenketten). In jede Tabelle wurden 250.000 zufällige Datensätze eingefügt. Um diese Testdatenbank mit insgesamt 1,25 Millionen Datensätzen zu füllen, wurde ein Tool entwickelt. Dieses Tool erstellt für jedes der Datenbanksysteme einen Thread, welche dann Transaktionen von 10.000 Datensätzen erstellen und diese in ihre jeweilige Datenbank einspielen. So entstehen für jede Datenbanktabelle 25 Transaktionen. Das Tool gibt außerdem Informationen zu dem Fortschritt aus. Einen Ausschnitt daraus ist in Abbildung 2 zu sehen.

```

file:///N:/Pro_Tec/_Allgemeine/Consulting Tool Demo/LargeDatabaseCreator/LargeDatabaseCreator/bin/Debug/LargeD...
11:48:19 - Start des Programms
11:48:19 - Der SQLite Thread wurde gestartet
11:48:19 - Der Access Thread wurde gestartet
11:48:20 - Verbindung zur Access Datenbank geöffnet
11:48:20 - Mit der Befüllung der 1. Tabelle der Access Datenbank wurde begonnen
11:48:20 - Verbindung zur SQLite Datenbank geöffnet
11:48:20 - Mit der Befüllung der 1. Tabelle der SQLite Datenbank wurde begonnen
11:48:28 - 1. Transaktion von 10000 Datensätzen für die 1. Tabelle der SQLite Datenbank erstellt
11:48:28 - Die 1. Transaktion der 1. Tabelle der SQLite Datenbank wird ausgeführt
11:48:28 - 1. Transaktion von 10000 Datensätzen für die 1. Tabelle der Access Datenbank erstellt
11:48:28 - Die 1. Transaktion der 1. Tabelle der Access Datenbank wird ausgeführt
11:50:19 - Die 1. Transaktion der 1. Tabelle der Access Datenbank wurde erfolgreich ausgeführt
11:50:27 - 2. Transaktion von 10000 Datensätzen für die 1. Tabelle der Access Datenbank erstellt
11:50:27 - Die 2. Transaktion der 1. Tabelle der Access Datenbank wird ausgeführt
11:52:24 - Die 2. Transaktion der 1. Tabelle der Access Datenbank wurde erfolgreich ausgeführt
11:52:33 - 3. Transaktion von 10000 Datensätzen für die 1. Tabelle der Access Datenbank erstellt
11:52:33 - Die 3. Transaktion der 1. Tabelle der Access Datenbank wird ausgeführt
11:55:34 - Die 3. Transaktion der 1. Tabelle der Access Datenbank wurde erfolgreich ausgeführt
11:55:47 - 4. Transaktion von 10000 Datensätzen für die 1. Tabelle der Access Datenbank erstellt
11:55:47 - Die 4. Transaktion der 1. Tabelle der Access Datenbank wird ausgeführt
11:58:37 - Die 4. Transaktion der 1. Tabelle der Access Datenbank wurde erfolgreich ausgeführt
11:58:47 - 5. Transaktion von 10000 Datensätzen für die 1. Tabelle der Access Datenbank erstellt
11:58:47 - Die 5. Transaktion der 1. Tabelle der Access Datenbank wird ausgeführt
12:01:58 - Die 5. Transaktion der 1. Tabelle der Access Datenbank wurde erfolgreich ausgeführt
12:02:06 - 6. Transaktion von 10000 Datensätzen für die 1. Tabelle der Access Datenbank erstellt
12:02:06 - Die 6. Transaktion der 1. Tabelle der Access Datenbank wird ausgeführt
12:04:15 - Die 6. Transaktion der 1. Tabelle der Access Datenbank wurde erfolgreich ausgeführt
12:04:22 - 7. Transaktion von 10000 Datensätzen für die 1. Tabelle der Access Datenbank erstellt
12:04:22 - Die 7. Transaktion der 1. Tabelle der Access Datenbank wird ausgeführt
12:06:26 - Die 7. Transaktion der 1. Tabelle der Access Datenbank wurde erfolgreich ausgeführt
12:06:33 - 8. Transaktion von 10000 Datensätzen für die 1. Tabelle der Access Datenbank erstellt
12:06:33 - Die 8. Transaktion der 1. Tabelle der Access Datenbank wird ausgeführt
12:07:29 - Die 1. Transaktion der 1. Tabelle der SQLite Datenbank wurde erfolgreich ausgeführt
12:07:39 - 2. Transaktion von 10000 Datensätzen für die 1. Tabelle der SQLite Datenbank erstellt
12:07:39 - Die 2. Transaktion der 1. Tabelle der SQLite Datenbank wird ausgeführt
12:08:39 - Die 8. Transaktion der 1. Tabelle der Access Datenbank wurde erfolgreich ausgeführt
12:08:47 - 9. Transaktion von 10000 Datensätzen für die 1. Tabelle der Access Datenbank erstellt
12:08:47 - Die 9. Transaktion der 1. Tabelle der Access Datenbank wird ausgeführt
12:10:33 - Die 9. Transaktion der 1. Tabelle der Access Datenbank wurde erfolgreich ausgeführt
12:10:40 - 10. Transaktion von 10000 Datensätzen für die 1. Tabelle der Access Datenbank erstellt
12:10:40 - Die 10. Transaktion der 1. Tabelle der Access Datenbank wird ausgeführt
12:12:26 - Die 10. Transaktion der 1. Tabelle der Access Datenbank wurde erfolgreich ausgeführt
12:12:34 - 11. Transaktion von 10000 Datensätzen für die 1. Tabelle der Access Datenbank erstellt
12:12:34 - Die 11. Transaktion der 1. Tabelle der Access Datenbank wird ausgeführt

```

Abbildung 2: Das Befüllen der riesigen Testdatenbank

SQLite wurde in Abbildung 2 wiederum mit grün und Access mit rot markiert. Access benötigt im Schnitt rund 2 Minuten um eine Transaktion mit 10.000 INSERT INTO Befehlen auszuführen. Für die gleiche Aufgabe benötigt SQLite ganze 19 Minuten.

Auf diese Datenbank wurden sowohl komplexe als auch relativ einfache Abfragen ausgeführt. Komplexe Abfragen, welche Joins von Tabellen, Filter, Sortierungen und Gruppierungen enthalten, konnten allerdings nicht zu Ende geführt werden, da ihre Ausführung mehrere Minuten gebraucht hat und sie nach ungefähr 15 Minuten abgebrochen wurden. Um trotzdem die Leistung der beiden Datenbanksystem vergleichen zu können, wurden hier zwei simple Abfragen ausgewählt und deren Leistung analysiert. Diese beiden simplen Abfragen sind in den folgenden beiden Code Listings zu sehen. Die dazu gehörigen Ergebnisse sind in Abbildung 3 und Abbildung 4 zu sehen.

1 `SELECT * FROM Tabelle1;`

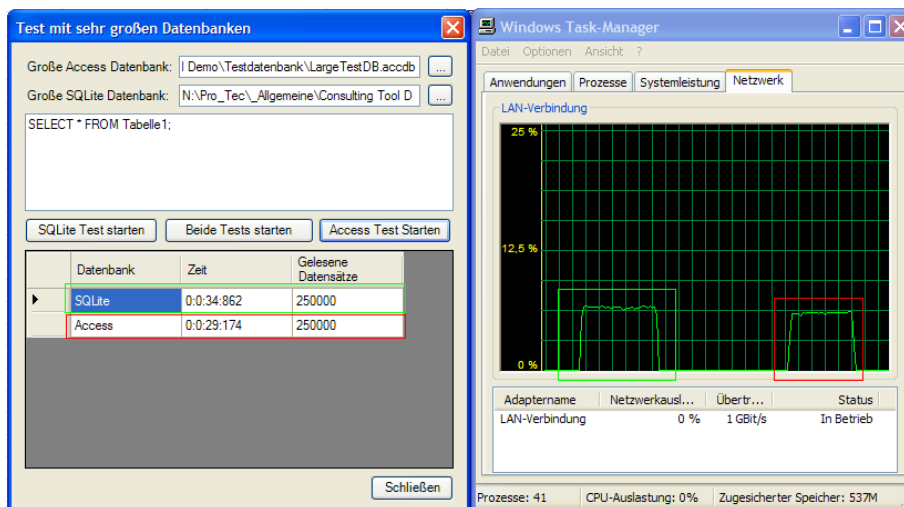


Abbildung 3: Selektieren von Daten in einer sehr großen Datenbank

Bei dem Selektieren aller Datensätze aus Tabelle 1 war Access fast 6 Sekunden schneller als SQLite (siehe Abbildung 3, linkes Fenster) und die Netzwerkauslastung der beiden fast war gleich (siehe Abbildung 3, rechtes Fenster).

1 `SELECT COUNT(*) FROM Tabelle1;`

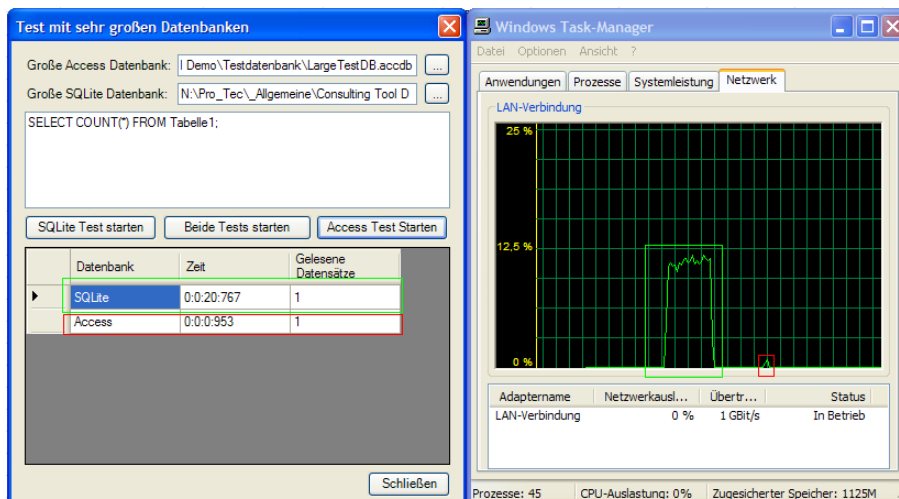


Abbildung 4: Zählen von 250.000 Datensätzen

Beim Zählen der Datensätze in Tabelle 1 gibt es allerdings einen großen Unterschied (siehe Abbildung 4). Wohingegen Access lediglich 0,953 Sekunden benötigt, braucht SQLite ganze 20,767 Sekunden. SQLite ist auch fast für die gesamte Netzwerkauslastung verantwortlich. Der Unterschied kann dadurch erklärt werden, dass Access anscheinend die Menge der enthaltenen Datensätze in der Datenbank speichert, wohingegen SQLite tatsächlich alle Datensätze liest und zählt, bevor es ein Ergebnis liefern kann.

Vor- und Nachteile

In der folgenden Tabelle werden die Vor- und Nachteile der beiden Datenbanksysteme zusammengefasst.

| Kategorie | Access | SQLite |
|---------------------------------------|--------|--------|
| Geschwindigkeit beim Lesen | + | - |
| Geschwindigkeit beim Schreiben | + | - |
| Größe der Datenbankdateien | - | + |
| Installationsaufwand | - | + |
| Integration in .NET | + | + |

Wie der vorherige Leistungsvergleich gezeigt hat, war Access im Vergleich zu SQLite immer schneller, sowohl beim Lesen und Auswerten der Daten als auch beim Schreiben der Daten. Nachteile sind bei Access allerdings die großen Datenbankdateien, welche teilweise sogar dreimal so groß sind wie eine vergleichbare SQLite Datenbank.

Ein Plus für SQLite ist der Installationsaufwand. Für SQLite muss lediglich eine Dynamic Link Library, zusammen mit der Software ausgeliefert werden. Diese muss nicht installiert werden, sondern kann einfach kopiert werden. Um mit einer Access Datenbank arbeiten zu können muss clientseitig der Access Datenbanktreiber installiert sein. Hierfür gibt es mehrere Möglichkeiten. Wenn Access installiert ist, so ist auch der Datenbanktreiber installiert. Dabei muss allerdings darauf geachtet werden, dass mindestens die gleiche Version von Access installiert ist, welche zum Erstellen der Datenbank verwendet wurde. Wenn ein Teil von Office installiert wird (in welchem Access nicht enthalten ist) kann der Datenbanktreiber ebenfalls mitinstalliert werden. Falls dies nicht der Fall ist, so muss der Datenbanktreiber nachinstalliert werden. Dieser wird von Microsoft auch als einzelne Installationsroutine frei zur Verfügung gestellt.

Bei der Integration in das .NET Framework sind beide Lösungen gleichwertig, da beide perfekt mit ADO.NET zusammenarbeiten.

Das größte Manko der beiden Datenbanksysteme ist, dass bei der Ausführung komplexer Abfragen auf große Datenmengen (siehe Abschnitt Leistung) die Leistung dramatisch einbricht. Laufzeiten von mehreren Minuten sind dann keine Seltenheit mehr.

Multi-User-Management

Wenn dateibasierte Datenbanken auf einem Netzwerklaufwerk liegen und über ein Netzwerk betrieben werden, so ist ein Multi-User-Management nur eingeschränkt möglich. Nach unserer Erfahrung können ca. 10 User gemeinsam mit einer Access-Datenbank arbeiten. Hierbei bietet Access ein einfaches User Management und sperrt temporär einen Tabellenbereich (Pages). Der Anwender wird beim Speichern darauf aufmerksam gemacht, dass ein anderer Anwender parallel Daten geändert hat. SQLite sperrt keine Tabellenbereiche. Mehrere Nutzer können gemeinsam Tabellenfelder editieren. Nur die letzte Änderung wird gespeichert. Auf der SQLite Website wird auf ein mögliches Fehlverhalten bei File-Locks hingewiesen, welches bei einer häufigen Benutzung der

Datenbank mit mehreren Clients dazu führen kann, dass eine Datenbank korrupt wird. Dies kann erfahrungsgemäß auch bei einer MS-Access Datenbank vorkommen. Hier empfiehlt sich ein regelmäßiges Reparieren und Komprimieren der Datenbank.

FAZIT:

MS-Access hat in unserem Performancevergleich besser abgeschnitten als SQLite. Die Stärken von SQLite liegen in seinem nicht vorhandenen Installationsaufwand und im Speicherplatzverbrauch der Datenbankdatei. Für das eingangs beschriebene Einsatzszenario, in welchem große Datenmengen verarbeitet werden müssen, sind beide Datenbanksysteme ungeeignet, da ihre Leistung bei komplexen Datenmengen/Abfragen einen dramatischen Einbruch erlebt. Laufzeiten von mehreren Minuten sind keine Seltenheit. Hier muss dann der Entwickler versuchen, die Abfragen hinsichtlich Ausführungsgeschwindigkeit zu optimieren. Ob dies gelingt, hängt vom Einzelfall ab.

Autoren:

Stefan Zeeb,
David Neumann

Kontakt:

th data GmbH
Poststraße 4
10178 Berlin

www.thdata.de